Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

EMAIL FRAUD CLASSIFIER USING MACHINE LEARNING

Vitor Carvalho

Director: Jordi José Bazán
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 13 de setembre de 2020

Abstract

Email is one of the most common methods of communication nowadays. Programs known as *malware detection* are essential to assist and protect users from the agents that are usually responsible for cyberattacks. This paper focuses on using *machine learning* algorithms to detect any possible email attacks by analyzing datasets of whitelists and blacklists. This document also includes other methods that try to solve this problem.

Contents

1 Introduction 5
2 Literature Review 7
2.1 Defining Fraud Email 7
2.1.1 Spoofing Attack 7
2.1.2 Phishing 7
2.1.3 Email Spam 8
2.1.4 CEO Fraud - Business Email Compromise (BEC) 8
2.1.5 Bonus Point – Social Engineering 9
2.2 Email Filters – Conceptual 9
2.2.1 List-Based Filters 9
2.2.2 Blacklist 9
2.2.3 Whitelist 10
2.2.4 Greylist 10
2.2.5 Real-Time Blackhole List 10
2.2.6 Content-Based Filters 11
2.2.7 Word-Based Filters 11
2.2.8 Rule-Based Filtering 11
2.2.9 Paul Graham's Bayesian Filtering 11
2.3 Email Filters – Practice 12
2.4 Machine Learning 13
2.4.1 Supervised Learning 15
2.4.2 Unsupervised Learning 15
2.4.3 Reinforced Learning 15
2.5 Biological Neural Network 15
2.6 Artificial Neural Network 16
2.6.1 Activation Function 17
2.6.2 Cost Functions - Loss Functions 17
2.6.3 Neural Network Architectures 18
2.7 Text Processing 18

2.7.1 Data Preprocessing 18
2.7.2 Natural Language Processing 19
2.7.3 Vectoring Data 20
2.8 Deep Learning 20
2.8.1 Forward Propagation 21
2.8.2 Back Propagation 22
2.8.3 Learning Algorithm 22
2.8.4 Core Problem in Deep Learning 24
2.9 Long Short-Term Memory (LSTM) 24
2.9.1 Recurrent Neuronal Network 25
2.9.2 LSTMs Structure 25
2.10 Summary 26
3 Data Collection 27
3.1 Fraudulent Email Corpus 27
3.2 URL Dataset 28
3.2.1 TF-idf Term Weighting 28
3.3 Summary 29
4 Methodology 30
4.1 TensorFlow 30
4.2 Scikit-learn 30
4.3 Email Classifier - Classic Approach 31
4.4 Email Classifier – Deep Learning 32
4.4.1 Creating the Model 32
4.4.2 The Train Subset 33
4.4.3 The Test Subset 34
4.4.4 Predict Phase 34
4.4.5 Sigmoid Function 35
4.5 URL Classifier 35
4.5.1 Logistic Regression 35
4.5.2 Algorithm Analysis 36

4.6 Email Classifier – Naïve Bayes 36
4.6.1 Bayes' Theorem 37
4.6.2 Multinomial Naïve Bayes 37
4.6.3 Code Overview 38
4.7 Summary 39
5 Analysis 40
5.1 LSTMs Results 40
5.2 URL Classifier – Results 40
5.3 LSTMs vs. Multinomial Naïve Bayes 40
5.4 Summary 41
6 Conclusion 42
References 43

1 Introduction

As the Internet started to be more common in the early 2010s, it was quickly recognized as an excellent tool to be used to attack. Anyone can email thousands of people just by a click. These emails can be classified as a malicious intent of cyberattacks. It is crucial to detect it and be prepared to avoid these kinds of contents. The user needs to understand how malicious agent attacks can be because this problem is severe since thousands of companies depend on these email tools to survive.

The most common way to avoid it is by reading the email content and checking the sender. If the user knows ahead of time which emails can have a malicious intention and which are not, they can waste less time reading and deleting. Big companies already identify and filter it, but still difficulty since the agent can use *social engineering* to catch the victim's attention.

There are many ways to avoid these kinds of attacks by using well-established techniques. This paper presents a tool that can help understand and filter a malicious email by using *machine learning,* more specifically, the *Long Short-Term Memory* subfield.

Big Data algorithms can learn by using datasets of dirty emails and ham emails. This process can be used to improve operations, provide better customer service, increase security, and much more. Companies that utilize big data hold a potential advantage over those that don't. Using the technique described in this document, the company or user can save time reading thousands of emails. And consequently, companies can save money in the process.

My main goal is conscious of how important it is to be aware of how much corrupt agents want our information. Sometimes we as an individual aren't the primary target, but the company we work for can be reached by ourselves. My other analysis consists of a brief look into the algorithm to automate the process of detection by using whitelists and blacklists. This method involves keeping a list of email addresses and either accepting (for a whitelist) or blocking (for a blacklist) email from addresses on that list.

Another thing that is important to keep in mind is that the datasets can change over the pass of time. The way the attack agent attacks can change, and even use different approaches that an insufficient dataset won't cover. It is essential to keep the dataset updated to obtain better results because a lot of factors can change it, such for example, the language of the emails, the email length, etc. Also, each dataset should be personalized with the user's needs. Said that it is essential to get with the idea and the results. The dataset used in this study won't cover the newest attacks tool.

The workflow followed by Big Data specialist starts with a question, in our case, how to detect fraud emails using machine learning. Secondly, get and clean the data, after that perform EDA (Exploratory Data Analysis). Thirdly, apply Big Data techniques. And finally, share insights and results. The previous workflow is the approach this document is going to follow. We are going to start with concepts and terminology, after that algorithms, and finally show the results and how to apply it.

In the meantime, the code implementation will use a couple of libraries. Libraries take care of all the hard work. So, we take care only of the focus on conceptual work. It means this document focus on the theoretical and lets the libraries work for us. The implemented tries to follow the *clean code* approach.

In the first section of this document, we are going to look at the machine learning process and algorithm and some other definitions, starting by defining what fraud email is. In Chapter 3, we describe the email structure and introduction to the datasets used. Then in Chapter 4, the method used to classifier an email by using the algorithm described in Chapter 2. Chapter 5 is a brief analysis of the results. And eventually, section 6 contains my conclusions and future works.

2 Literature Review

Before we start with, it is essential to study the definitions. In this chapter, we show how we define fraudulent emails, and we also offer some methods that are used to prevent it. Additionally, we are going to look at machine learning techniques and how they can be used to avoid this problem. Also, there are definitions of biological neural networks, text processing, and much more.

2.1 Defining Fraud Email

Before one can get into the machine learning algorithm, an accurate definition of malware must first be provided. Email fraud or also called email scam is the process of intentionally deceiving for personal gain or damaging another individual or company through email. There are a lot of methods used by harmful agents to archive such a goal. Let's take a look at some of them.

2.1.1 Spoofing Attack

A spoofing attack is known as an email sent from someone or a program pretending to be someone else. This method can be through a phone call, websites clone, emails, or using more advanced techniques such as Domain Name System (DNS). This type of attack is possible because the core email protocols do not have any mechanism for authentication.

The bad agent can pretend to be someone from the company or a friend and ask for personal data. To be credible, it uses simple words or emotional expressions. Machine learning algorithms can be trained to detect these behaviors and patterns.

Email spoofing has been responsible for public incidents with severe business and financial consequences. For example, an attack at Medidata Solutions in 2017 that caused a $4.8 million loss. The scam started with a spoofed email that was sent to an employee purportedly from Medidata's president, directing the employee to wait for an attorney's wire transfer instructions to pay for an impending acquisition. That same day, the purported attorney called with instructions to process the wire transfer, and a subsequent spoofed email induced both Medidata's vice/president and its CFO to sign off on the assignment. The company realized that it had been defrauded two days later. More details can be found in the reference.

2.1.2 Phishing

Phishing scams are the most popular and thus dangerous form of email fraud. Harmful agents use email messages that appear to come from a legitimate company, institution, or someone we know, such as your bank, university, or your coworker, and ask your personal information to “update” or “verify” your personal information.

These corrupt agents can also use a chain email that consists of a form of junk email. A chain mail message is generally sent to several people and includes instructions that each person should forward the letter to several others. This type of attack can waste system resources or collect data from these followed steps.

The first phishing lawsuit was filed in 2004 against a young boy who created a copy of a popular website. He was able to obtain sensitive information from users and access credit card detail to withdraw money from the website users.

There are standard features or flags of phishing emails, such as a sense of urgency, hyperlinks, unusual sender, attachments, and eye-catching statements. These features provide specific patterns and can be used in email classifiers.

2.1.3 Email Spam

People describe email spam as directly “unsolicited commercial email sent in bulk”. Hormei created the name SPAM in 1937 as the world’s first canned meat that didn’t need to be refrigerated. The word started being used on email after a TV show, Monty Python’s Flying Circus. In the skit, a group of Vikings sings “SPAM, SPAM, SPAM” repeatedly. This sketch started a new trend, and they called started calling SPAM everything that was annoying.

In the air 1979, Tom Truscott and Jim Ellis at the University of Duke started calling the annoying messages received in their network spam.

Most of the spam messages are just annoying, but some of them can also be dangerous because they can contain links to phishing web sites or sites that are hosting malware.

The most popular anti-spam techniques include email filtering based on the content of the email, DNS-based blackhole lists, greylisting. Each method has strengths and weaknesses. In the next section, we are going to explain some of them.

2.1.4 CEO Fraud - Business Email Compromise (BEC)

As mentioned before, the lousy agent can pretend to be the CEO of a company, usually requesting to the finance department for money transfer or send confidential information. The fake email usually requires an urgent answer to minimize investigations and skepticism.

This type of attack is sometimes associated with spoofing. A sophisticated scammer might also be stalking the company’s CEO personal accounts. Learning from his online fingerprints and when he is out of the office or on holiday, the bad agent attacks the company.

Machine Learning can spot this kind of fraud and other spear-phishing attempts by learning the organization's specific communication patterns and can detect anomalies based on factors that escape the notice of human users. Over time, the AI gets better at spotting it, as we are going to see in the following chapters.

2.1.5 Bonus Point – Social Engineering

The spoofing, phishing, and CEO Fraud emails are all parts of social engineering, which is where use tricks and tactics to obtain benefits. This information could be used to gain information about the company for a future attack.

Social engineering attacks, besides being common against enterprises and users they're also increasingly sophisticated. All platforms are vulnerable to suffer this type of attack since the user is the key to it. A social engineering email is designed to look like it is from a credible organization, like your message service, a delivery company, or even a bank.

2.2 Email Filters – Conceptual

There are many ways to go about solving the problem of filtering emails. This section describes several methods that can be used to solve this problem and automate this process.

2.2.1 List-Based Filters

This method can be used along with other methods because it requires only a list of uses to categorize senders as attackers or trusted users and to block or to allow their messages accordingly.

It is the simplest method and can also be used with other methods. The list can grow over time to keep it safer and updated.

2.2.2 Blacklist

Blacklist is a popular method to filter email. It is kind of like the previous process mentioned. The administrator creates a list of senders that are used for blocking messages. Blacklists are a list of Internet Protocol (IP) addresses or email addresses that have been previously used to send fraud emails. The way it works is checking to see if its IP or email address is on the blacklist when an incoming message arrives; if so, the message will be rejected.

Harmful agents use blacklist on both large and small scales. A third party would usually provide a large-scale blacklist. The user typically does not contribute to an extensive list like this. On a smaller size, the user could simply tell their email client not to allow email from specific addresses. A small-scale blacklist is useful if the user only gets a fraud email from a particular address.

2.2.3 Whitelist

A whitelist blocks possible attacks by using a system almost exactly opposite to the blacklist method. It allows the user to specify which senders to allow emails from, instead of blocking them. Whitelist methods can be beneficial because it could be used along with another filtering method.

This method can also be a problem since it is impossible to predict who is going to send an email. Any sender that is unknown to the user will be filtered out. The previous problem can be avoided by checking the blocked emails and accepting the ones that aren’t threatening emails.

2.2.4 Greylist

This method uses the fact that many spammers or harmful agents only attempt to send a batch of junk email once. Under the greylisting system, the receiving mail server initially rejects messages from unknown sources and sends a failure message to the originating server. When the email is sent a second time, the greylist assumes the message is a valid email and lets it proceed to the recipient’s inbox. Also, the recipient’s email or IP address is added to the allowed senders.

This method is not recommended if the users expect some messages to be delivered quickie because it can cause delays in some messages.

2.2.5 Real-Time Blackhole List

This method is like the blacklist but with less hands-on maintenance since it is maintained by third parties, who take the time building a backlist using data collected by their users.

Real-time blackhole List is a service where a simple DNS query mail check if a sending IP address is on a blacklist of IP addresses reputed to send fraud emails. This software usually contains many list resources to flag these emails as fraud or not.

2.2.6 Content-Based Filters

The previous methods mentioned use policies for all messages from an email or IP address. This method filters messages by evaluating words or phrases found in each message to determine whether an email is a fraud or legitimate.

The method uses in this document can be considered a content-based filter since it studies the words used in a whitelist and blacklist to classifier it as a legitimate mail or not.

2.2.7 Word-Based Filters

Word-Based Filters is the simplest type of content-based filter, and they simply block any email that contains specific terms.

It is common to see some patterns in junk email, like “You won 1000€, send us your information so you can claim it”. Also, tempt words can be used to catch the receiver’s attention. This method can filter it by checking common words.

The inconvenience of word-based filters is that it can cause false positives. An excellent example of it is emails that contain the word “discount.” Maybe the users are interested in getting some discounts, but the filters reject these kinds of emails.

2.2.8 Rule-Based Filtering

Rule-based (or heuristic) filters take things a step beyond simple word-based filters. Rules are defined to classify emails as fraudulent or ham based on different characteristics. An example could be that all emails with a specific color text are fraudulent. Another example would be that emails that contain “order confirmation” is a ham email.

The combination of rulers can also be used to make it a better decision. As mentioned before, the emails follow some patterns that can be used to classifier it.

The harmful agents are doing their best to make their messages look credible. Because of that, it is common to receive fraud emails that contain ham patterns. Because of that, it is hard to create rules that will work well in all cases.

2.2.9 Paul Graham’s Bayesian Filtering

Paul Graham (1964) is a computer scientist and programmer. He is best known for his work on the programming language Lisp. In august of 2002, he wrote *A Plan for Spam,* in which he discussed why rule-based filtering fails to filter spam effectively. The paper proposed a new method of spam filtering using Bayer’s rule.

A Bayesian filter is considered the most advanced form of content-based filtering, employ probabilities to determine which messages are legitimate and which aren't.

The similarity with the method studied in this document is the fact that Bayesian filters learn. To decide the probability that an email is considered fraudulent based on the words it contains, the filter needs to know about the emails that the user usually receives. In his document, he uses the headers of the message. Unfortunately, the dataset used doesn't have the message headers. As a result, it can cause different effects and less accuracy. It is essential not to ignore data when using machine learning and Bayesian Filters in a real situation.

Theoretically, Bayesian filters become more effective the longer it is used since they are building they word list based every time an individual user reives a message. However, the users need to train it first by deleting the junk messages.

Fraud Email and ham emails are kept in separate hash tables. So that probabilities can be calculated later. When an email is declared fraudulent, the fraud table is updated by incrementing the frequency counts for each word that the email contains.

Graham has suggested using a modified Bayes' rule to calculate probabilities. This method, when combining multiple possibilities, is

$$P(A|B \cap C) = \frac{P(A|B)P(B|A \cap C)}{P(B|C)}$$

Graham's modification of Bayes rule is:

$$P(A|B \cap C) = \frac{P(A|B)P(A|C)}{P(A|B)P(A|C) + (1 - P(A|B))(1 - P(A|C))}$$

The meaning of $P(A|B \cap C)$ is the probability of event A be true, given that events B and C are correct. So, it can be used in email filtering by asking the likelihood of an email A be fraudulent, while B and C correspond to certain words being in that email.

By applying this formula, we can assume the formula for each word:

$$P(Fraud|word) = \frac{\frac{b}{nbad}}{\frac{g}{ngood} + \frac{b}{nbad}}$$

Where g is the number of times that word appears in the right email and b how much times it appears on a wrong email. "*ngood*" and "*nbad*" are the total numbers of ham and fraudulent emails received by the user, respectively.

2.3 Email Filters – Practice

The email transportation process can be represented, as shown in figure 1. Each process is compatible with spam filters. This technique can occur on both server sides, the clients and the senders. Also, hardware like a router can block email. In this document, the focus in on the client-side, more specifically in the user’s machine. But, the techniques in this document can also be implemented on the server-side.

As tools, we are going to use Python and some third-party libraries. Python works great in any machine and has a lot of tools that can make the project more comfortable, such as TensorFlow and Keras. We are going to look at these in the next chapter.

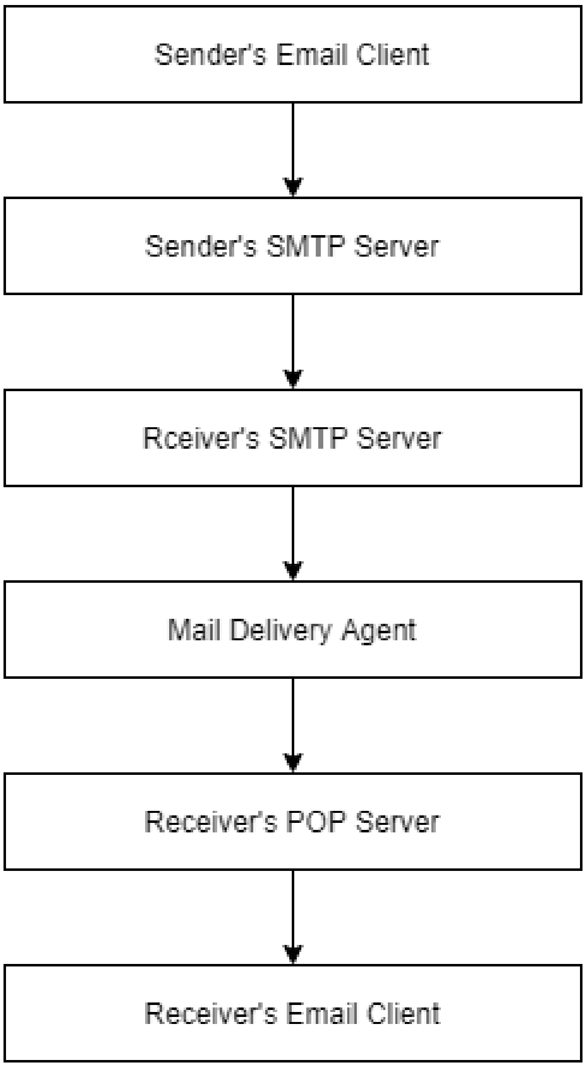


Figure 1: Email transport process

2.4 Machine Learning

Our brain uses connections between neurons in the brain to process and transmit information. Such a thing consists of the neural circuit. The human biological neural networks are the main inspiration for the design of artificial neural networks. In this section, we show how machine learning methods work and when to use them.

Before anything else, it is vital to understand the difference between *Artificial Intelligence*, Neural Network, and Machine Learning.

Intelligence is logical to make reason, ability to learn, plan, and solve problems. Artificial Intelligence is the effort to automate intellectual tasks usually performed by humans.

Neural Network mimics the way the human brain operates with a series of algorithms that aim to recognize underlying relationships in a set of data.

Machine learning is a subset of artificial intelligence associated with creating algorithms that can change themselves without the programmer intervention to get the desired result, as we can see in figure 2. The algorithm feeds itself by analyzing structured data. In other words, it is the science of programming computers so that they can learn from data.

Here is a slightly more machine learning general definition:

"*Machine Learning: Field of study that gives computers the ability to learn without* being explicitly programmed", Arthur Samuel (1959)

"*Well posed Learning Problem: A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*" Tom Mitchell (1998)

There are three types of machine learning algorithms, the supervised ones, the unsupervised ones, and reinforced learning.

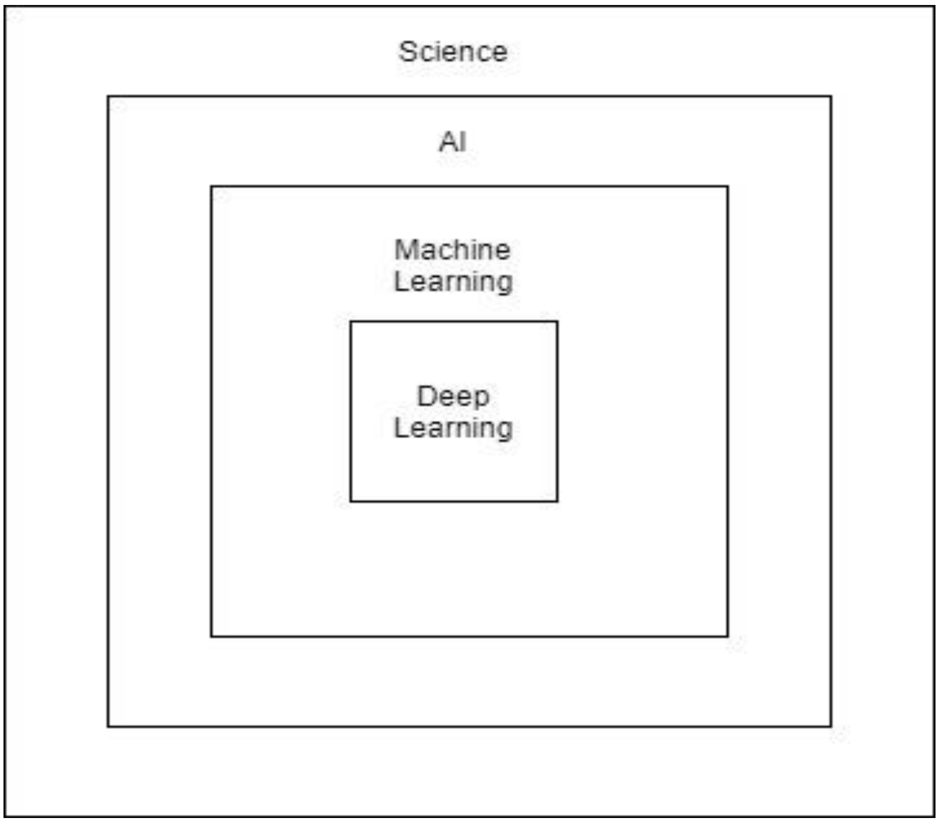


Figure 2: Machine Learning Scheme

2.4.1 Supervised Learning

Algorithms are designed to learn by example. The idea is that there is a relationship between the input and the output. The machine knows each result should look like, and the machine learning algorithm adjusts itself to find a connection between the output and the input.

We feed the algorithm with the features associated with labels. The machine predicts the rulers and returns them. These rulers can be used to label new features. In our case, we can use it to determine if the email has a fraudulent character or not.

A typical supervised learning task is classification—that why it is perfect for building an email filtering. The email dataset contains each email classification (fraud or ham). Each example is a pair consisting of Input Objects (typically a vector) and the Desired Output.

2.4.2 Unsupervised Learning

If we have little or no idea what our results should output, we can use the Unsupervised Learning. We still have the features, but we don't have any labels for those features. The model returns the tags for these features. In other words, the system tries to learn without a teacher.

Unsupervised learning is used in a task that detects features and relationships in the data. For example, identifying unusual behavior in our bank account or study patterns on the visitors of a website. The list can go on and on.

2.4.3 Reinforced Learning

Reinforced Learning is a very different approach to the last two techniques. This method uses an agent that explores an environment to learn the best way how to solve the problem. Each movement or action it does have a reward. This reward can be archived by doing something right; otherwise, this agent receives a penalty, so he knows if his actions can lead to his objective or not.

Reinforced Learning is accessible nowadays because it is used to solve games since games tend to give awards and penalties based on the player's actions.

2.5 Biological Neural Network

Even though neural networks are very far from the real biological neural network, the connections between computing elements in artificial neural networks are modified by learning rules based on the nervous system. In this subsection, we are going to look at it.

A *Neural Network* (NN) is a network or circuit of artificial neurons or nodes. These neurons are connected using weight. A Neuron is a cell and the principal component of the nervous system in which the primary functions are to receive, process, and send the information via the biological system. It is divided into three parts: soma, dendrites, and axon.

The soma is the non-process portion of a neuron, containing the cell nucleus. In the meantime, the dendrites are branched protoplasmic extensions of a nerve cell that propagate the electrochemical stimulation received from other neural cells to the cell body, or soma. Electrical stimulation is transmitted onto dendrites by upstream neurons (usually via their axons) via synapses, which are located at various points throughout the dendrites tree. And eventually, the axon typically conducts electrical impulses known as action potentials away from the nerve cell body.

2.6 Artificial Neural Network

Artificial neural networks (ANN) or connection systems are computing systems that are based on the biological neural network mentioned in the previous section. These systems emulate a physical neural network to “learn” to perform tasks by considering examples. In our case, the whitelist and blacklist, see chapter 3 for more detail. This section is designed to study ANN’s main concepts.

ANNs are at the very core of Deep Learning and comprise three layers: input, hidden, and output. They are versatile and can scale quickly, making them ideal for tackling large and highly complex Machine Learning tasks.

An ANN consists of a collection of connected units or nodes called artificial neurons (known as either a neurode or perceptron). Each connection, like the synapses in a biological brain, can transmit a signal to another neurode. This neurode has structures like our neurons. Like the cell, the dendrites are the channels that allow input (x_1, x_2, ..., x_n). A function processes this input and equivalent to the soma of the biological neuron cell. And last, the output channel (y) is the computer representation of the axon.

These connections provide the output of one neuron as an input to another neuron. Each relationship is assigned a weight that represents its relative importance. The mathematical representation is:

$$y = \sum_{i=0}^{n} w_i x_i + b_i$$

2.6.1 Activation Function

The activation Function introduces Non-Linearity in the network and decides whether a neuron can contribute to the next layer. In a nonlinear system, the change of the output is not proportional to the evolution of the input.

The output of each perceptron there is the activation function. This function transforms the value of the result of this perceptron. Later this value is passed to another artificial neuron. Typically, this transformation involves using a nonlinear function, such as a sigmoid, hyperbolic-tangent, and others.

Activation functions also help normalize the output of each neuron to a range between 1 and 0 or between -1 and 1. In new neural networks, it is also applied to the Backpropagation to train the model, which places an increased computational strain on the activation function. In Image 3, we can see the representation of this process.

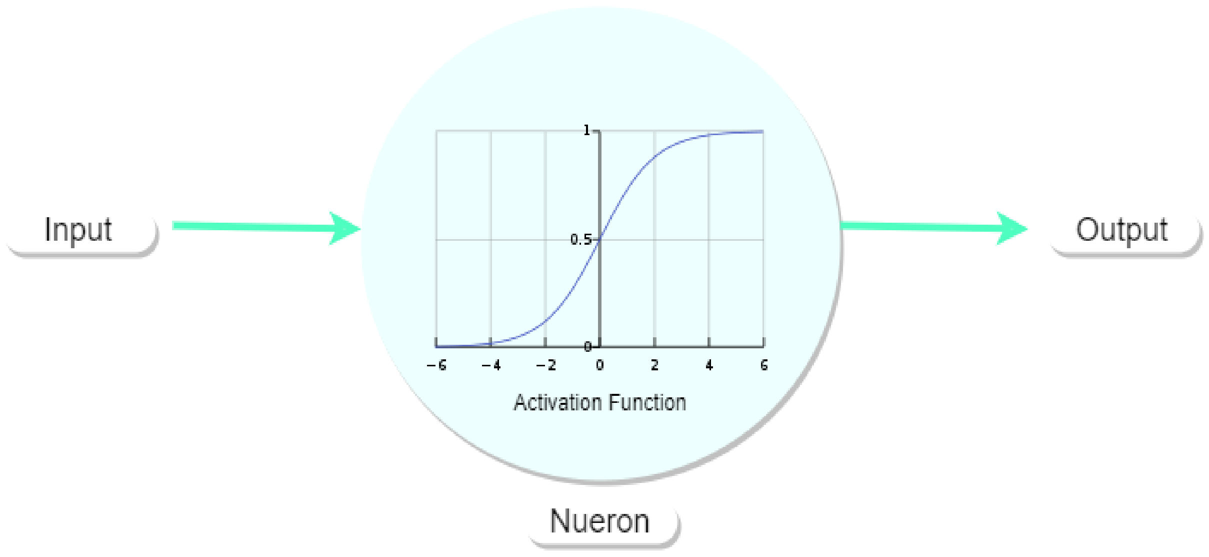


Figure 3: Activation Function inside the neuron

2.6.2 Cost Functions - Loss Functions

To evaluate the performance of an artificial neuron, the ANN uses the cost function. The cost function measures the distance between the estimated value that the neuro has and the real value

The variables used in the cost function:

- y := real value
- a := estimated value by the neuron
- w := weight between two connected neurons
- x := value received by the neuron

- b := residual value or Bias
- z := value that we pass to the activation function to calculate the estimated value, “a”

$$z = wx + b$$

2.6.3 Neural Network Architectures

There is a different type of neuronal network architectures. Let’s start with the Fully Connected Feed Forward Neural Networks in which each neuron in the proceed layer is connected to every subsequent layer with no backward connections, no cycles, and no loops in the core. However, the more neurons it is added, the computational resources to training the model increases.

Recurrent Neural networks or Feed-Forward Neural Networks take in fixed-sized inputs and returns fixed-sized outputs. Recurrent Neural Networks uses a feedback loop in the hidden layers. This process uses the state of the previous data to make new predictions. Because it can remember the past, this architecture can’t model every modern problem. It is common to see this architecture used in natural language processing, sentiment analysis, DNA sequence classification, speech recognition, and language translation.

Convolutional Neuron Networks (CNN) is a type of neural network designed for specific tasks like image classification. It was inspired by the organization of neurons in the visual cortex of the human brain. Like the Fully Connected, CNNs are composed of the input layer, the output layer, and several hidden layers.

2.7 Text Processing

Before we start to get deep in the code and methodology of our machine learning, we need to process our text. This method is necessary because the algorithm handles differently from us humans. We need to feed the text in a way the algorithm can operate as a structured format. A structured layout has a well/defined pattern, whereas unstructured data has no proper structure. Let’s look at how to preprocess our text.

2.7.1 Data Preprocessing

First, the most important thing to do is convert sentences to words and remove unnecessary information, like punctuations, familiar words, words with no meaning. For example, when we process the URL, it eliminates the URL extension, to be sure it won’t be processed unnecessarily. This process is known as cleaning the data.

Removing the punctuation of a sentence makes it meanness to us humans, but for the machine is the other way. The device doesn't need to study the context of each word. For example, "Good morning?", the person who said that maybe think it is evening, but he just woke up, so he is saying it is expecting to be morning. The machine won't give this meaning to this sentence.

Secondly, it is essential to Tokenizer the text. This method consists of separating the text into units such as sentences or words. For example, "Today is a good day" -> ["Today", "is", "a", "good","day"].

Finally, we remove common words or stopwords; these words appear in any text. They don't add much meaning.

After the preprocessing is done, we can do two more steps to clear more our text called stemming and lemmatizing. Stemming a word helps reduce it to its stem form. We want our terms that mean the same to be treated equally. For example, "Working" has the suffices "ing" and we want to be processed as "work". Lemmatizing helps to transform a word into its canonical form ('lemma'), i.e., the root forms, such as the word "plays" to "play".

2.7.2 Natural Language Processing

NLP (Natural Language Processing) is a field in machine learning with the ability of a computer to processing the information to manipulate it. Natural language processing can be separated into two parts: natural language, such as English and Portuguese, and handling. Processing is defined as how computers carry out instructions. To simplify how computers deal with textual data.

In Python, we can use NTLK (Natural Language Tool kit) o/and Keras. NTLK is a popular open/source package that provides all everyday NLP tasks.

In this document, we use both libraries. It is a deep learning library that provides essential tools to help prepare text data.

It provides a sophisticated API called Tokenizer for preparing text that can be fit and reused to make multiple text documents. It is perfect for our project since it returns all information about our data:

- *word_docs:* it is a dictionary of words and how many records each word appeared in.
- *word_counts*: it is a dictionary of words and their counts.
- *word_index:* it is a dictionary of words and their uniquely assigned integers.
- *document_count:* it is an integer count of the total number of documents that were used to fit the Tokenizer.

Once the Tokenizer has been fit on training data, it can be used to encode text in the train or test datasets. For this task, we are going to use the function *texts_tot_matrix(),* which provides a suite of standard bag-of-words model text encoding schemes that can be delivered via a mode argument to the function. We selected count, and it does what says, counts the words in the document.

2.7.3 Vectoring Data

After the preprocessing and processing is done, the data need to be vectorized. Vectorizing data is the process of encoding text as an integer. The numeric form of an encoded-word helps create feature vectors so that machine learning algorithms can understand our data. There is a lot of methods to do it. In this document, we use one-hot encoding.

Each word has a definite value, which represents the numerical value of that word in the dataset. Similar concepts have the same categorical amount. This value starts at 0 and goes all the way up to N-1, where N is the number of unique words. That way, each email has a list of categorical values.

2.8 Deep Learning

Let’s take a look at Deep Learning, which is a subset of machine learning where algorithms are created and function similarly to machine learning. Still, there are many levels of these algorithms, each providing a different interpretation of the data it conveys. It resembles the neural connections that exist in the human brain. This section is designed to study the Deep Learning algorithm and fundamental definitions.

Deep Learning allows the computer to build complex concepts out of simple ideas. Why learn Deep Learning? Deep learning models have revolutionized many areas of machine intelligence, such as:

- Speech recognition
- Natural language processing
- Image recognition
- Handwriting recognition
- Text processing

In other words, Deep Learning is a Machine learning technique that learns features and tasks directly from data. Data Inputs run through “neural networks”. The neural networks have hidden layers. Train themselves to understand patterns in the data and Output useful predictions. A simple neural network contains three necessary layers: the input layer, the output layer, and several hidden layers.

The Deep Learning process can be broken into two primary methods: Forward Propagation and Backpropagation.

2.8.1 *Forward Propagation*

The input layer is a set of neurons that are connected to the next layer using channels that have a numeric value called a layer. The output of the layers is processed using the activation function, and the final layer, the result, has an absolute probability value.

$$y = \sigma(\sum_{i=1}^{n} w_i x_i + b_i)$$

The activation function is shifting to the right or to the left according to the Bias, and the weight represents how vital is that neuron.

The information goes from the input layer to the output layer. The input is several neurons, x_i to x_n. These neurons are connected to the neurons of the next level through channels. Each channel has a numerical value called weight, and it is multiplied by the input value, and the Bias is added as well. Figure 4 represents this process.

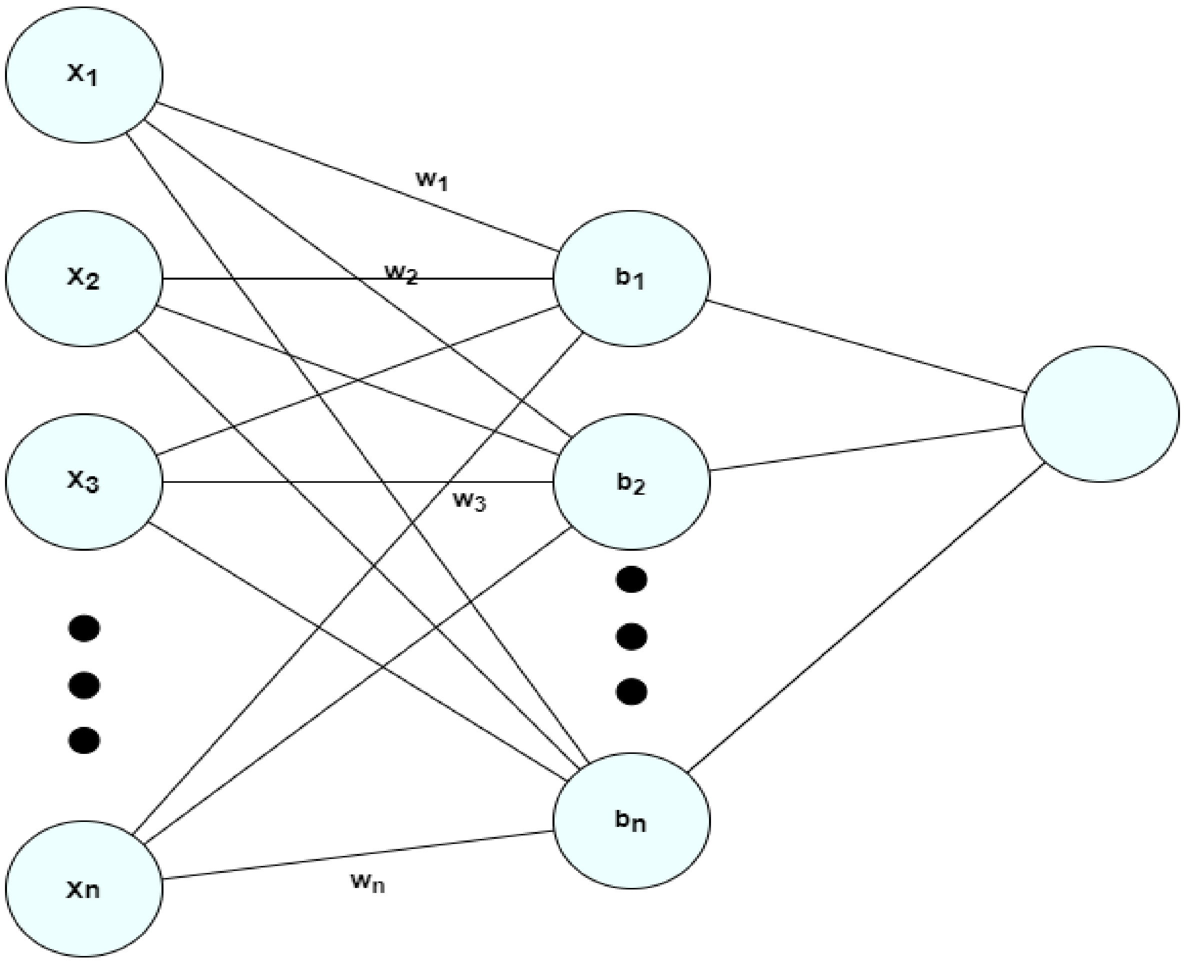


Figure 4: Forward Propagation Graph

2.8.2 Back Propagation

Back Propagation is the Forward Propagation but reversed; in other words, the information is passed from the output layer through the hidden layers. It is the reason the Deep Learning process is so powerful and can learn by itself because it adjusts the initial weights and biases using a Loss Function. The values adjusted is used to fit the prediction model.

Loss Functions help quantify the deviation of the predicted output by the neural network to the expected result—consequently, the accuracy of the algorithm increases. There are plenty of loss functions, for example, under Regression, Squared Error, Huber Loss. In Binary classification: binary cross-entropy, Hinge Loss. And under Multi-Class Classification: Multi-Class Cross-Entropy, Kaulback Divergence. Different projects require different loss functions.

2.8.3 Learning Algorithm

After the “neuronal network” goes through all the data, it is now able to make predictions, the Learning Algorithm. This process can be summarizing in the follow’s steps:

1. Initialize parameters with random values
2. Feed input information to the network
3. Compare the predicted value with the expected value and calculate the loss
4. Perform Back Propagation to propagate this loss back through the network
5. Update parameters based on the loss
6. The last step is to continue integrating previous actions till loss is minimized

During the training, we adjust the parameters to reduce the loss function and make our model as optimized as possible. Tie together the loss function and model parameters by updating the network based on the output of the loss function.

The Loss Function guide optimizers. For example, the Gradient Descent is an iterative Algorithm that starts at a random point on the loss function and travels down its slope in steps until it reaches the lowest point of the function.

How does it work? First, calculate what a small change in each weight would do to the loss function. Secondly, adjust each parameter based on its gradient and finally repeat steps 1 and 2 until the loss function is as low as possible.

When dealing with High dimensional Datasets, where can have a lot of variables, it is possible to be stuck in a Local Minimal. In Gradient Descent, we ideally want to find the global minimum of the loss function. To avoid getting stuck in a local minimum, we use the proper Learning Rate.

We use a small number to represents the Learning Rate. This number is multiplied to scale the gradients, and it can be, for example, 0.001. We don’t want a large learning rate, where the algorithm will skip the global minimum. Similarly, we don't want one too small either, which can cause the algorithm to take forever to converge to the global minimum. Figure 5 shows the visual representation of the gradient descending.

To ensure efficiency, we use Stochastic Gradient Descent. Like the Gradient Descent, except uses a subset of the training examples rather than the entire lot. It is an implementation of the Gradient Descent that uses batches on each pass using momentum to accumulate gradients.

When the dataset is large, we need to look at some terminologies: epochs, batches, batch sizes, and iterations.

One epoch represents when the entire dataset is passed forward and backward through the neural network only once. Multiple epochs are used to help the model generalize better. The more the number of epochs increases, the better our parameters get. A large number of epochs can be harmful because it can cause the predictions to memorize the previous values. The right amount of epochs depends on the dataset.

It is better to divide large datasets into smaller batches and feed those batches into the neural network. The total number of training examples in a Batch is the Batch Size. The Iterations are the number of batches that need to complete one epoch.

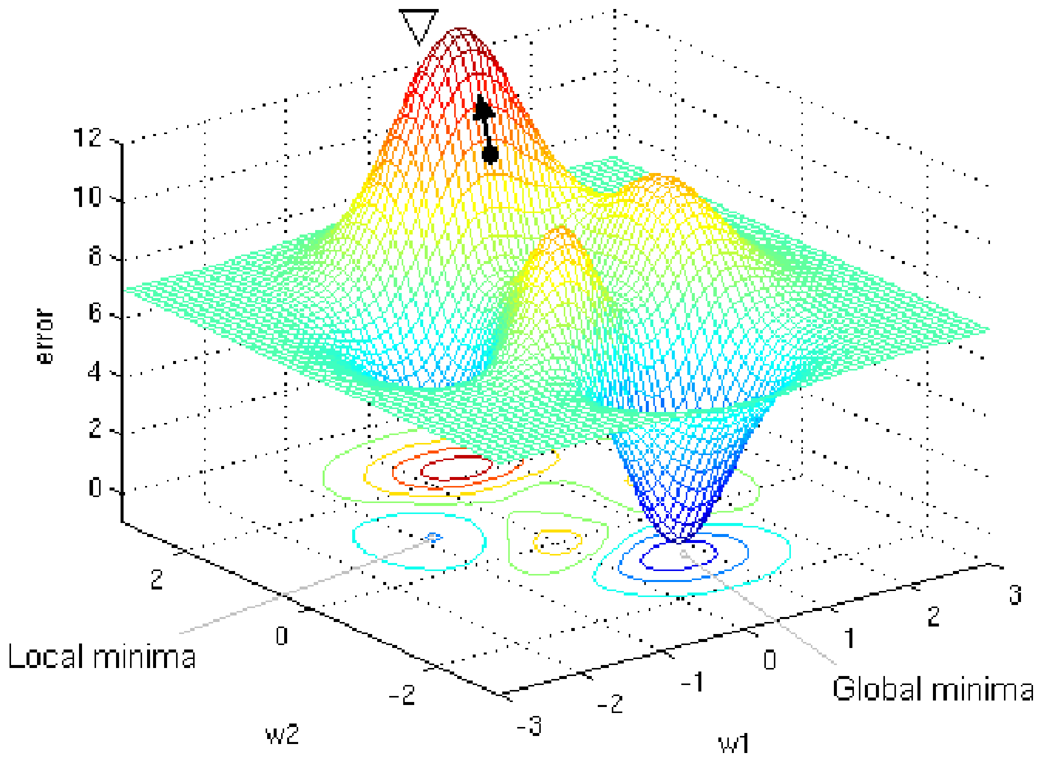


Figure 5: Gradient Descending

2.8.4 Core Problem in Deep Learning

When the model isn't performing well on training data and new test data, it causes Overfitting, which happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data.

There are some methods to detect Overfitting: Dropout, Augmentation, Early Stopping, and Cross-Validation. The dropout method randomly removes some nodes and their connections, resulting in different results and adding randomness to the algorithm to predict better. The dataset Augmentation adds fake data to make the model better. Early stopping consists of stopping the model training when the validation error increases after a certain point. The Cross-Validation splits the training dataset into multiples training sets and validations sets.

Dataset Augmentation is better for classification algorithms because it applies transformations on the existing dataset to synthesize more data. For example, rotating or scaling images. Early stopping is the most used due to its simplicity and effeteness.

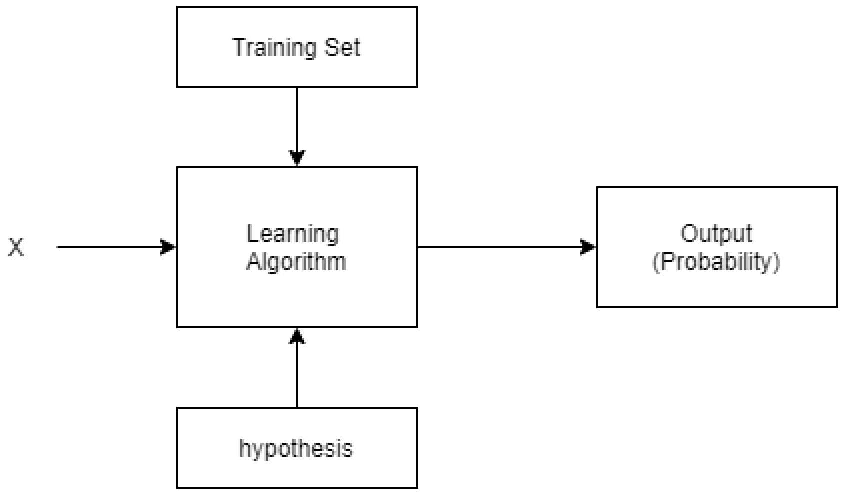


Figure 6: Learning Process

2.9 Long Short-Term Memory (LSTM)

LSTM is an artificial recurrent neuronal network (RNN) architecture used in the field of Deep Learning. LSTMs is the structure used to train the principal model. In this subsection, we are going to study the RNN and LSTM.

2.9.1 Recurrent Neuronal Network

The RNN is much more capable of processing sequential data such as text or characters than other neuronal network architectures. When we say recurrent neural networks, all we mean is a network that contains a loop. An RNN will process one word at a time while maintaining an internal memory of what it's already seen like humans do when we read a text. Our brain remembers previous words and sentences to give the text a context—that why it is perfect to work with Natural language processing.

Each input is treated based on the context of the previous data. All networks based on recurrent neural networks have the form of a chain of repeating modules of the neural network allowing information to be passed from one step of the system to the next. Each module in the standards RNN contains a single tanh layer.

2.9.2 LSTMs Structure

Long Short-Term Memory networks – usually just called "LSTMs," are a special kind of Recurrent Neuronal Network capable of learning long-term dependencies.

Since it is based on RNN, LSTMs also have a chain-like structure; the difference is that the repeating module has four that interact between them instead of having a single neural network layer.

Each LSTM cell maintains a cell state vector, and at each time step, the next cell can choose to read from, right to it, or reset the cell using an explicit gating mechanism. Each unit has three binary gates of the same shape: input gate, output gate, and the "forget gate".

The input gate controls whether the memory cell is updated, the "forget gate" control if the memory cell is reset to 0 (stands for forgetting the information). The output gate controls how data is visible to the next section. All of them have a sigmoid activation function.

A part of these three gates, the LSTM cell, has another vector that modifies the cell state with tanh function activation. Tanh distributes gradients. Hence prevents the information from vanishing or exploding.

Each of the gates takes the hidden state and the current input X as inputs. It concatenates the vectors and applies a sigmoid. C_t represents a new candidate value that can be used to the cell.

The C_t is the only vector that can modify the cell state. The forget gate controls how much of the old state should be unremembered. This state is applied to the output gate to get the hidden vector.

The step-by-step of the LTSMS start with the “forget gate”. It looks at the previous output, h_{t-1}, and the new input, x_t, to output a number between 0 and 1. The mathematical representation is the following:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

The next step consists in deciding which value is going to be stored in the cell. First, it determines which value update:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Secondly, a tan layer creates a vector of new candidates values that could be added to the state:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

After that is time to update the old cell state, C_{t-1}, into the new cell state, C_t, by multiplying both states:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

The last step is deciding which value is going to be the output of the cell. First, we run a sigmoid layer, which determines what parts of the cell state we’re going to produce.

$$o_t = \sigma(W_0 \cdot [h_{t-1}, x_t] + b_0)$$

And then the cell state is pushed to be between 0 and 1 by using a tanh function:

$$h_t = o_t * \tanh\,(C_t)$$

LSTMs look intimidating. However, the process of building it is kind of simple. In chapter 4, we show how to make it using TensorFlow.

2.10 Summary

Build an email filtering is not easy as it may seem. Almost every method has its downside, and users can still lose important messages if the technique doesn’t work correctly. Machine Learning isn’t different; the users need to be aware that even if the methods seem to work fine, the harmful agents can also use techniques to make their email look credible.

In this section, we had taken a look at several definitions that are important to understand what is the problem we are trying to solve and how to solve it. Also, we have seen that to make sure our Deep Learning code works is essential to process the data before inserting it into our model.

3 Data Collection

Machine learning algorithm needs large amounts of data to train, especially text models. In our case, we're going to use emails in text format, pre-classified as ham, or fraud email. It is essential to keep these emails pure. By pure, we mean that the ham email dataset doesn't contain threatening emails, and fraud emails only contain bad emails.

Our collection must come from different sources. Since email datasets are kind of expensive, we are going to work with a public dataset, and hopefully, we can reach our primary goal. The section below describes the process it was used to collect the datasets. Also, it contains the analysis and how to process these datasets. As mentioned in the previous section, most classification methods require that features be encoded using simple value types, such as integers, booleans, and floating. All the datasets used can be found on Kaggle.

3.1 Fraudulent Email Corpus

Enron Corporation, based in Texas, was an energy, commodities, and services company that went bankrupt in 2001. The company was founded in 1985 when Houston Natural Gas and InterNorth merged. It was one of the largest independent developers and producers of electricity in the world. Nobody was expecting such a big company to run into difficulties.

At the end of 2001, it was revealed that Enron's reported financial condition was sustained by a traditional, systematic, and creatively planned accounting fraud. By the year 2002, Joe Barting, litigation support and data analysis contractor working for Aspen Systems, collected an extensive database of over 600 000 emails generated by 158 employees of the Enron Corporation.

This dataset is the primary source of data for our email classifier. Most of the analysis is done using NLTK and Sklearn libraries. The first thing we are going to look at is the word frequency distribution from NLTK to get top words and their use counts. To make this step more visible, we are going to use another library, WordCount. This library allows us to Matplot and seaborn for formatting and display options. In figure 7 is shown the result of the plotting the most common words.

Each row of this dataset is an email and his classification, 1 for a fraud email, 0 otherwise. We won't change these classifications, since the category is already a numerical value.

The process used in this dataset was the following:

1. Open the CSV using pandas and drop the duplicates values.
2. Clean the text using the process described in subsection 2.7.

3. We are vectorizing the data, in our case, using the function one hot encoding mentioned in the subsection Text Processing.

After the process is finished, each email is a vector that contains only numerical values. Each uncial word has a unique integer value. These vectors have different sizes, which isn't suitable for our model. The function pad_sequences() from the Keras library help us to make sure our lists have a length of 255 values.

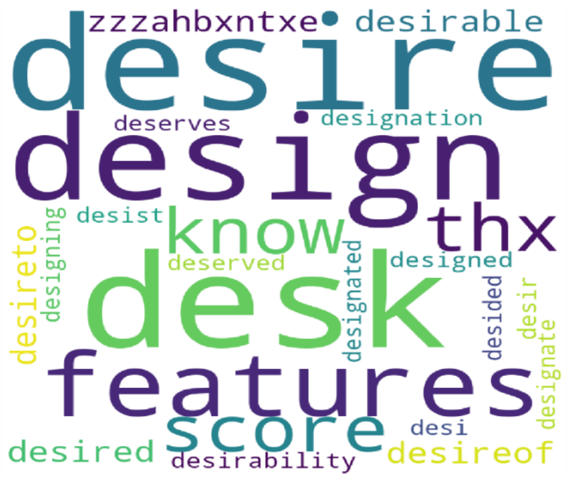


Figure 7: Top 25 most used words in the Enron's emails dataset.

3.2 URL Dataset

One of the first approaches of this study was to analyze the URLs that emails contain. A simple URL can tell a lot about the email. During my research, I found a URL dataset that I can work to build a URL classifier. In this dataset, there are classifications of a trustful URL and a list of bad URLs that I wouldn't insert my credentials in.

The process to Tokenizer the URL is different from the method used to Tokenizer the emails. For each URL in our dataset, we separate it into words by splitting it by dash, dot, slash, and hyphen. Each URL has a domain, and we remove it from them. These URLs are going to use to train a small model to detect if a URL is good or bad. The process of building can be found in the next chapter.

The method used to encode the data is the TF-IDF term weighting.

3.2.1 TF-IDF Term Weighting

As mentioned before, the raw data, a sequence of symbols, cannot be fed directly to the training algorithms. These algorithms expect numerical feature vectors with a fixed size as most large documents will typically have words that will be very present. So, it is crucial to balance the weight of these words.

To re-weight, the count features into floating-point values suitable for usage by our classifier. For each word in the document, we calculate an inverse proportion of the frequency of that term to the percentage of the world manifests itself in the text.

3.3 Summary

From all the sources mentioned before, it was collected thousands of emails and URLs. The messages and URLs contain the classification.

In this section, we also analyze the email dataset, which was already cleaned and only included the messages and also included each email classification. A brief history of the Enron Company is provided as well.

4 Methodology

There are many ways to build a deep learning algorithm using Python. When I decided to work on this project, I was curious about how machine learning works, and I decided to use TensorFlow as the main framework.

This section is going to focus on the logic behind the source code. First, we are going to look at the libraries used in the project. Secondly, we show a brief look at how to build the main code. But before that, there is an analysis of each mathematical function or expression used on it.

At the end of this chapter, I also include an alternative solution to our problem that requires less system but also powerful. Besides, this section discusses the theory behind the Naïve Bayes classifiers, the logistic Regression mathematical fundaments.

This section contains pieces of code. We provided the entire code in the src folder. This document is just a guideline to understand the central keys in the code.

4.1 TensorFlow

TensorFlow is an open-source platform that makes machine learning accessible. It provides a lot of tools that can be used to build any kind of algorithm to make machines learn.

TensorFlow was started in 2017 by Google Brain to create machine learning systems based on deep learning neural networks. Nowadays, it is the most popular machine learning platform with thousands of users.

This open-source platform can run on multiple GPUs and CPUs deepening on the main objective. TensorFlow uses Distribution Strategies to run on multiple GPUs. In our project, we use it with GPU since it is faster than the CPU. By default, TensorFlow maps almost all the GPU memory visible to the process.

Why use TensorFlow? Tensorflow is built to facilitate the creation and deployment of ML models. Machine Learning is helping software to automate tasks without explicit programming rules. In traditional programming, the developer specifies how the program reacts to the inputted data. In machine learning, the mindset is different. We provide examples, and the computer learns the patterns to create a solution. Tensorflow offers several levels of abstraction, so the developer chooses the best suits his model.

4.2 Scikit-learn

Scikit-learn is a well documented free software machine learning library for Python and contains a lot of well-implemented functions for machine learning and statistical modeling. It features various algorithms, such as classification, Regression, supervised models, and clustering. It also includes support from other Python libraries.

This powerful library started at Google by David Cournapeu is used to build a multinomial classifier that we are going to compare with our deep learning algorithm. It can be used as a simple and efficient tool for predictive data analysis.

4.3 Email Classifier - Classic Approach

Let’s start with the traditional approach of email filtering:

1. First, we read the email. While reading it, we start our flags. These flags are variables we analyze before writing the program, like some words or phrases, such as “credit card”, “free”, “last chance”, “reply asap”. Besides, the flags also detect the email structure, colors, and so on.
2. The detection algorithm evaluates the flags stated before and would produce an output, such as a Boolean, true for the wrong email, false otherwise.
3. The final step is accepting the email or report it as fraud attention.

The next figure is a graphical visualization of the classic approach:

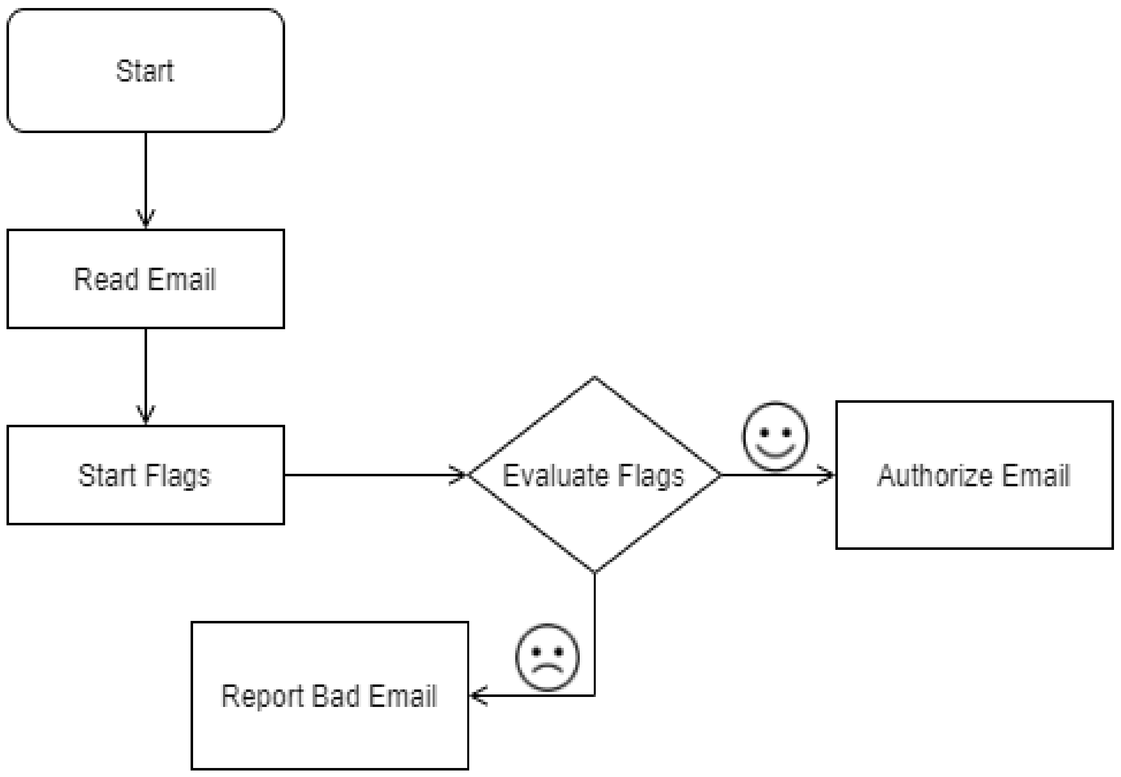


Figure 8: Traditional approach of email filtering

You might think that not hard, but when you start to code it, you miss a lot of things you didn’t consider. And the code is a mess because it is just a list of complex rules and if statements. Moreover, if the lousy agent notices some words that he is using can be one of our filters, he would start using another word. Consequently, we would have to change our flags every time this happens.

In contrast, an email filtering using Machine Learning algorithms detects these rules for us, and much more we would miss. The code is easy to maintain and gets more accurate with time.

4.4 Email Classifier – Deep Learning

In chapter 3, we showed the Enron's emails dataset where each email is labeled. The labels are 1 (stands for fraud) or 0 (stands for not fraud email). Using this dataset, we can build classifiers based on supervised learning that will automatically tag a new email using the labels mentioned before. First, we read and process our dataset, as discussed in section 3.1.

The next step is to separate the dataset in two big piles, the test, and the train subsets. The separation can be done using the function *train_test_split()* from the sklearn library. This function split arrays or matrices into the random train and test subsets. After separating the dataset, our next mission is to create the model.

It is essential to determine how trustworthy the model is and what purpose we can use it. The model evaluation tells us whether a classification model is accurately capturing a pattern.

4.4.1 Creating the Model

Working with a Deep Learning model requires five simple steps to archive results. The first step is Gathering data. In this step, it is essential to pick the right data. After the selection of the data, we can move to the next level, which is preprocessing the data. This step includes the randomly splitting the dataset into subsets. How to do it is showed int the Data Collection section.

Once data is prepared, we can proceed to step 3, which consists of feeding the data into the neural network to be used to training the model. This process is described in the Deep Learning subsection.

The fourth step consists of evaluating the model by assessing it in the real world. In this step, the model is to compare his predictions with the actual values to decide how accurate the system is. Both levels can be seen in Figure 9.

Now it is time to create the model. We'll use a word embedding layer as the first layer in our model and add an LSTM layer afterward that feeds into a dense node to get our predicted email.

The output dimension of the vectors generated by the embedding layer is 32. Also, the activation function used is sigmoid. In the following subsections, we are going to study it. The code below shows it is easy to create the model using TensorFlow:

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(total_words, 32),
    tf.keras.layers.LSTM(32),
    tf.keras.layers.Dense(1, activation="sigmoid")
])
```

And finally, the last step, optimize the model.

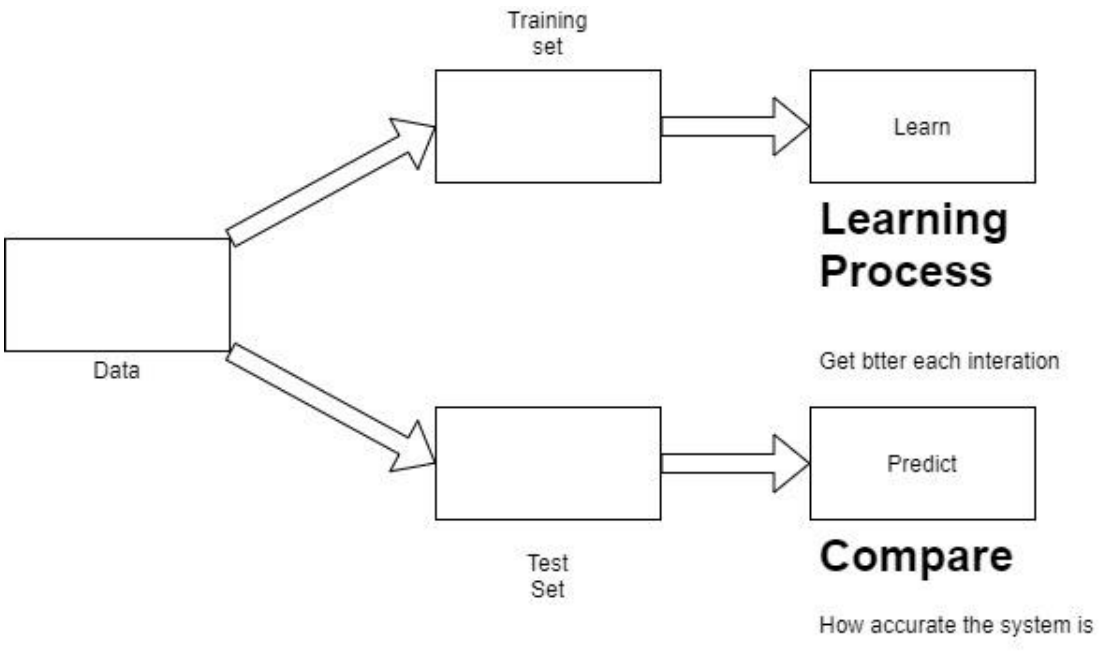


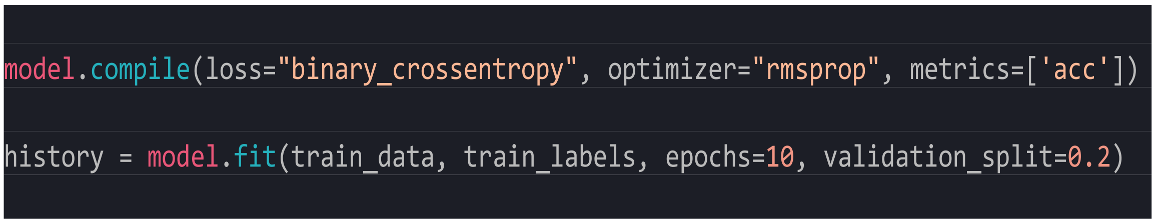
Figure 9: Learning Process and Compare the model

4.4.2 The Train Subset

Supervised learning is built based on a training dataset containing the correct label for each label input. The training set is used to train the model.

During training, a feature extractor is used to convert each input value to a feature set. These features set, which capture the necessary information about each that should be used to classify it. Pairs of feature sets and labels are fed into the machine algorithm to generate the model.

The code below train the model using the training dataset:



```
model.compile(loss="binary_crossentropy", optimizer="rmsprop", metrics=['acc'])

history = model.fit(train_data, train_labels, epochs=10, validation_split=0.2)
```

The number of epochs is not the same for different models. Some experts in the area don't know how much epochs and batches are necessary. They decide it by experience or testing the model. The more model we build.

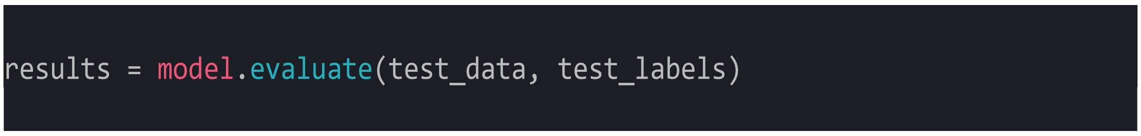
Training a model is a constant searching for the values that will fit the data better. That the reason the data is separated into two big piles, one to train the model and another to test it. In other words, we test if the number of epochs and batches are suitable for the model prepared.

4.4.3 The Test Subset

Evaluation techniques calculate a score for a model by comparing the labels that are generated for the inputs in the test subset with the correct tags for those inputs. And the test subset is the same format as the training subset.

It is essential to keep the test subset separate and unused in our model until it is completed training. During the model is training, the "test subset" is used to evaluate how well our model will perform on new input values. This process will give us the accuracy of the email classifier.

To evaluate the model, we just write the following code:



```
results = model.evaluate(test_data, test_labels)
```

4.4.4 Predict Phase

Once the model is trained and tested, it can be used to classifier new data. It is essential to process this new data the same way the dataset was processed. The output is a probability value for the entered data. This probability represents the percentage of an email be fraudulent or not. If the likelihood is higher enough, we can use it to reject the email that provided that message.

The model can also be stored in the user's machine for future predictions.

4.4.5 Sigmoid Function

A sigmoid function is a mathematical function which has a characteristic "S"-shaped curve or sigmoid curve. A wide variety of sigmoid functions have been used as the activation function of artificial neurons. The most common sigmoid function used in the neuronal network is the logistic function shown in the next subsection.

A sigmoid function is a bounded, differentiable, real function that is defined for all actual input values and has a non-negative derivate at each point and precisely one inflection point. A sigmoid "function" and a sigmoid "curve" refer to the same object.

4.5 URL Classifier

To build and URL classifier, we are going to use the URL dataset mentioned in the last chapter. The main library used for this task is the sklearn.

The idea is simple – given an URL never seen before, determine whether that URL is malicious or not. This task looks simple for us humans if you look to the URL and see a domain is never seen before or words like the words mentioned in Figure 3.

While this task is easy for us humans, it's much harder to write a program to a program classify it. Our script starts with reading our CSV using pandas.

We will be using *Logistic Regression* since it is fast and it is the most extensively used statistical technique for predictive modeling analysis.

In statistics, the logistic model is used to model the probability of a particular class or event existing, such as bad/good. Let's take a look at the Logistic Regression fundaments.

4.5.1 Logistic Regression

Logistic Regression is a way to explain the relationship between a dependent variable and one or more explanatory variables using a straight line. There are two types of linear Regression – Simple and Multiple.

The basic form of the logistic Regression uses a logistic function to model a binary dependent variable, even though you can find many more complex extensions. A logistic function is a familiar S-shaped curve with an equation:

$$f(x) = \frac{L}{1 + e^{-k(x \ \ -x_0)}}$$

Where x_0 is the value x of the sigmoid's midpoint, L is the curve's maximum value, and k the logistic growth rate or steepness of the curve.

The standard logistic function $\sigma(t) : \mathbb{R} \rightarrow (0, 1)$ is defined as follows:

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

By assuming that t is a linear function of a single explanatory variable. We can express t as $t = \beta_0 + \beta_1 x$. And now the logistic function $p(x) : \mathbb{R} \rightarrow (0, 1)$ can be written as:

$$p(x) = \sigma(t) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

In the logistic model, $p(x)$ is interpreted as the probability of the dependent variable y equaling to 1 (stands for success/case) rather than a 0 (positions for failure/non-case). Logistic Regression is used in a supervised learning classification algorithm to predict the probability of the target variable.

4.5.2 Algorithm Analysis

The first thing is to read the dataset and preprocess the data. For each URI in the dataset, we separate it in unique words/tokens and remove unnecessary data such as the domain. After preprocessing the data, the following step is vectorizer the tokens found. The vectorization is done using the function *TfidfVectorizer* provided by the library sklearn. Also, we separate the dataset into two piles, one for training and another for testing using another function provided by the sklearn library: *train_test_split*.

Once the dataset is ready for the training process, we can use the logistic regression classifier to train our model and make predictions.

4.6 Email Classifier – Naïve Bayes

Bayes' Theorem has been the main inspiration for a collection of classification algorithms, and Naive Bayes classifiers are one of them. Each algorithm of the groups of algorithms shares a universal principle. This principle is that every pair of features being classified is independent of each other. Naïve Bayes can perform surprisingly well in the classification tasks where the probability itself calculated by naïve Baes is not essential.

Text classifiers based on naïve Bayes have been studied and extensively applied in many applications. They work well in many real-world situations, especially in the spam filter field. In this document, we use Naïve Bayes methods that are a set of supervised learning algorithms. They require an amount of training data to estimate the parameters.

The data used is the same used in the Deep Learning algorithm to make it easy to compare both algorithms. Despite being extremely fast compared to more sophisticated methods, they are lousy estimator.

4.6.1 Bayes' Theorem

Bayes' Theorem states the following relationship, given a class variable y and dependent feature vector x_1 though x_n:

$$P(y\,|x_1,\dots,x_n) = \frac{P(y)P(x_1,\dots,x_n \mid y)}{P(x_1,\dots,x_n)}$$

Using the naïve conditional independence assumption that

$$P(x_i|y,x_1,\dots,x_{i-1},x_{i+1},\dots,x_n) = P(x_i|y)$$

Which we express as:

$$P(y\,|x_1,\dots,x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|\,y)}{P(x_1,\dots,x_n)}$$

Since the denominator is constant for a given input, we can remove it:

$$P(y\,|x_1,\dots,x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|\,y)$$

To create a classifier model, we find the probability of a given set of inputs for all possible values of the class variable y and select the output with maximum likelihood.

$$\hat{y} = \arg\max_{y} P(y)\prod_{i=1}^{n} P(x_i|\,y)$$

Where $P(y)$ is called class probability and $P(x_i|\,y)$ is called a conditional probability. The difference of each naïve Bayes classifier is the assumptions they make regarding the distribution in the conditional probability.

4.6.2 Multinomial Naïve Bayes

The scikit-learn library provides the MultinomialNB, which implements the naïve Bayes algorithm for multinomially distributed data and is one of the two classic naïve Bayes variants used in text classification.

Vectors parametrize the distribution $\theta_y = (\theta_{y1}, \dots, \theta_{yn})$ for each class y. N is the number of features, in our case, the size of vocabulary using the one-hot encoding.

The conditional probability, θ_{yi}, of i appearing in a sample belonging to class y. To maximizer the relative frequency counting, the conditional probability is estimated using the following formula:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_{yi} + \alpha n}$$

Where $N_{yi} = \sum_{x \in T} x_i$ represents the number of times i appear in a sample of class y in the training set T, and $N_{yi} = \sum_{i=1}^{n} N_{yi}$ is the total count of all features for class y.

4.6.3 Code Overview

The first thing we need to do is create a vector of features for training and testing. For comparing with our Deep Learning model, we followed the same steps mentioned in section 3.

Once the training and the testing set are prepared, we can start training the model. The model training is pretty easy to code, as we can see in the following code:

```
X_train, X_test, y_train, y_test = train_test_split(
    tokens, data['Class'],
    test_size = 0.2,
    random_state = 0
)

classifier = MultinomialNB().fit(X_train, y_train)

pred = classifier.predict(X_train)
```

The tokens are the emails coded using the one-hot encoding, and with the classification of each, we create the vectors for training and testing. The MultinomilaNB() function from scikit learn returns the model trained.

4.7 Summary

Machine learning models are more comfortable to build once you understand all the mathematics behind it. For example, LSTMs mathematics looks scary, but once you know it, the processing of building a model requires just call the right function.

In this section, we studied how to build our model and the phases implied in the model construction. Some technical explanations are also provided to explain each step.

5 Analysis

In this section, we are going to analyze the results of our models. It is essential to keep in mind that the results might change if we use different datasets and different machine learning models. The accuracy obtained is the result of feeding the test pile into the model. Since we already know the output, we can see if the predicted results are the ones expected.

5.1 LSTMs Results

Our email classifier did well for the dataset we have used. The estimation of the effectiveness of an LSTMs we use the test dataset. In our case, our model scored 0.969268 of accuracy, which means it can classifier a fraud email as a fraud email with an error of 3.07%.

The score is high for a simple RNN, but the reason for that is how clean our dataset is. The fraud ones don't contain any ham email, and the worlds used in it can be easily distinguished from words used in a ham email.

It is possible to use the model to classify single emails. We only need to encode it using the same function used to encode the dataset.

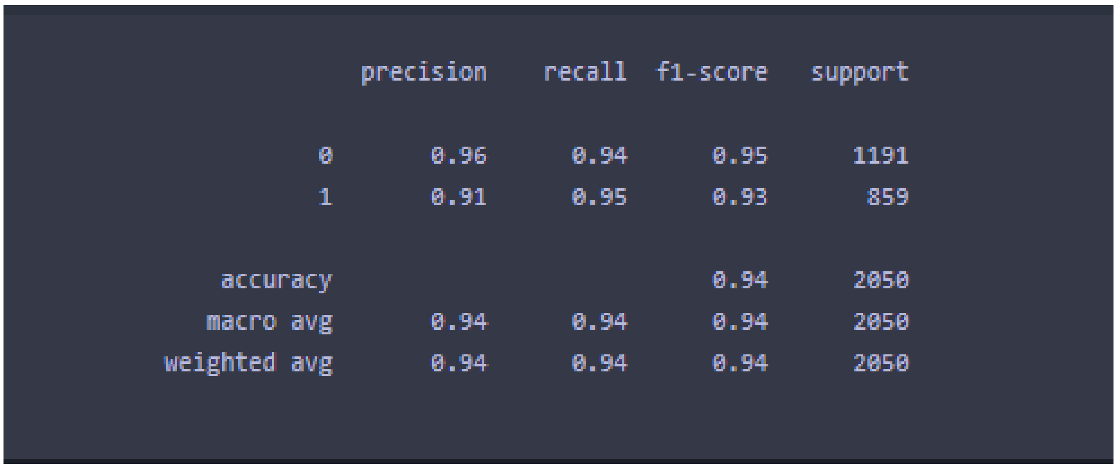
5.2 URL Classifier – Results

Logistic Regression is the most common Machine Learning approach and considered the "Hello world of Machine Learning" by many specialists in this field. Said that I decided to create my first Machine Learning algorithm using the approach mentioned before. The result is pretty good and somewhat similar to the previous one. The accuracy is 0.96182.

This model output good for an unthreatened URL and bad for threatening URL, which can be used alongside the previous model to create a better classifier. It can be used alone to prevent the user connect to the URL domain.

5.3 LSTMs vs. Multinomial Naïve Bayes

It is more interesting to compare different models, especially when they have other structures. Even though Naïve Bayes isn't good at text classification as the LSTMs, our result is outstanding. With a precision of 0.94, which is pretty close to 0.969268. The following table shows us the report of the Multinomial model:



```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1191
           1       0.91      0.95      0.93       859

    accuracy                           0.94      2050
   macro avg       0.94      0.94      0.94      2050
weighted avg       0.94      0.94      0.94      2050
```

In this table, we can see the reported averages, which include macro average (averaging the unweighted mean per label), weighted average (averaging the support-weighted standard per tag), and sample average (only for multi-label classification). The value 1 stands for fraud email, and the value 0 stands for a ham email.

As we can see, this model also archives our primary goal. It can detect a fraud email with outstanding accuracy. LSTMs and Naïve Bayers classifiers can solve this problem.

The difference is that the NB belongs to a category called generative, which implies that during training, the classifier tries to find out how the data was generated in the first place. It essentially tries to figure out the underlying distribution that produced the dataset input to the model. Consequently, Naïve Bayes classifiers are faster than LSTMs.

On the order hand, RNN is a discriminative model. Discriminative models try to figure out what the differences are between the positives and negatives in the dataset to perform the classification.

While Naïve Bayes classifies are well known and famous for decades, RNNs are starting to be the first preference for text classification. Due to his power to work using the GPU resources.

With the right amount of data and a few lines of code, it is possible to build a robust email classifier that requires less maintenance than code build following the traditional software development.

5.4 Summary

This section is designed to discuss the results of the models implemented, Logistic Regression, MultinomialNB, and LSTMs. Also, a bit of comparison between the Multibinomial and LTSMs is provided.

6 Conclusion

In this paper, we evaluate the viability of using Deep Learning techniques to solve the fraudulent attacks through emails.

The main objective of this study was to make a descriptive analysis of the technology implicated in building a Deep Learning email classifier. Our remarkable results are achieved with LSTMs, which works well for text processing. Since the model is generic, it can be used to train other datasets. This model applies in different fields; the data just need to be encoded using the methods described in this document or other methods suitable for natural language processing.

On the one hand, the equations behind the neuronal network look petty intimidating. On the other hand, once you understand it, the code implementation only requires a few lines of code because there are tons of functions to assist the process of building it from scratch.

In summary, our model can help us to understand linguistic patterns and can be used to make predictions in new similar data. One important thing to keep in mind is that the most common words our dataset are not words that will be used often to determine whether or not an email is a fraud because they are commons words like “the”, “you”, “and”, etc. Once the model is trained and tested, it is ready to classifier new data.

Another point to consider when building a text classifier is Naive Bayes techniques. Naive Bayes learners and classifiers can be speedy compared to more sophisticated methods; besides, they need a small amount of data.

There are a lot of future projects that could be done to extend our study. For example, the model can turn into an API. Another example can be collecting personal data and automated email filtering.

References

- Steven Walczak, Narciso Cerpa, in Encyclopedia of Physical Science and Technology (Third Edition), 2003
- TensorFlow: Biology's Gateway to Deep Learning, Ladislav Rampasek and Anna Goldenberg, 2016
- Source Coding: Part I of Fundamentals of Source and Video Coding, Vol. 1, No 1 (2011) 1–217, Thomas Wiegand and Heiko Schwarz
- Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems, Aurélien Géron
- Raizer, Klaus & Idagawa, Hugo & Nobrega, Euripedes & Ferreira, Luiz. (2009). Training and Applying a Feedforward Multilayer Neural Network in GPU.
- Anti-Spam Methodologies: A Comparative Study Saima Hasib, Mahak Motwani, Amit Saxena Truba Institute of Engineering and Information Technology Bhopal, 2012.
- Understanding LSTM Networks Posted on August 27, 2015, on Stanford: https://web.stanford.edu/class/cs379c/archive/2018/class_messages_listing/content/Artificial_Neural_Network_Technology_Tutorials/OlahLSTM-NEURAL-NETWORK-TUTORIAL-15.pdf
- Hafsa Jabeen, 2018 Stemming and Lemmatization in Python: https://www.datacamp.com/community/tutorials/stemming-lemmatization-python
- 7 Types of Neural Network Activation Functions: How to Choose: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/
- Enron Corporation - Company Profile, Information, Business Description, History, Background Information on Enron Corporation, 1400 Smith Street Houston, Texas 77002-7369, U.S.A: https://www.referenceforbusiness.com/history2/57/Enron-Corporation.html

- ¿Cuál es el origen de la palabra SPAM? Javier Cordero: https://www.javiercordero.com/origen-palabra-spam/

- Neural circuit from Wikipedia: https://en.wikipedia.org/wiki/Biological_neural_network

- Naïve Bayes, 2007 - 2020, scikit-learn developers (BSD License): https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes

- Natural Language Processing with Python, by Steven Bird, Ewan Klein, and Edward Loper, Copyright © 2019: https://www.nltk.org/book/ch06.html

- Recent Cases on E-Mail "Spoofing" Coverage Highlight the Impact of Specific Crime Policy Wordings, August 29, 2017, By Benjamin Duke, Matt Schlesinger, and Scott Levitt: https://www.insideprivacy.com/united-states/litigation/recent-cases-on-e-mail-spoofing-coverage-highlight-the-impact-of-specific-crime-policy-wordings/

- Common Phishing Scams: https://www.phishing.org/common-phishing-scams